Amendments to the Claims

Please amend Claims 1-24. The Claim Listing below will replace all prior versions of the claims in the application:

Claim Listing

1. (Currently amended) A method for circuit emulation over a multi-packet label switching (MPLS) network, comprising:

receiving a time division multiplexed data stream at an ingress end from an optical carrier;

dividing ~~said~~ the data stream into a set of fixed sized packets;

adding a service header to each of ~~said~~ the packets;

adding an additional header on top of ~~said~~ the service header in accordance with MPLS protocols;

removing ~~said~~ the additional header after each packet has been processed by ~~said~~ the MPLS network;

using ~~said~~ the service header to recover ~~said~~ the data stream at an egress end;

storing a first set of frames of the data stream into a data buffer during times clocks of the ingress and egress ends cannot be traced to a common reference source;

calculating a first data average of ~~said~~ the first set of frames in ~~said~~ the data buffer to obtain a threshold value;

storing a next set of frames into ~~said~~ the data buffer;

calculating a next data average of ~~said~~ the next set of frames in ~~said~~ the data buffer;

comparing ~~said~~ the next data average to ~~said~~ the threshold value; and

if ~~said~~ the next data average is greater than ~~said~~ the threshold value[[;]], generating a negative justification indicator[[;]] and sending one more byte at ~~said~~ the egress end[[;]], or if ~~said~~ the next data average is less than ~~said~~ the threshold value[[;]], generating a positive justification indicator[[;]] and sending one less byte at ~~said~~ the egress end.

2. (Currently amended) The method of claim 1, further comprising:

monitoring ~~said~~ the data stream; and

attaching an alarm bit in a service header of a subsequent packet if a break in ~~said~~ the data stream is detected.

3. (Currently amended) The method of claim 1, further comprising:

using a negative justification bit and a positive justification bit in ~~said~~ the service header to indicate whether ~~said~~ the synchronous payload envelope includes a negative stuff byte or a positive stuff byte.

4. (Currently amended) The method of claim 6 [[1,]] ~~further comprising:~~ wherein the structure pointer reserves ~~reserving~~ a pointer value indicating that ~~said~~ the header byte is not present within ~~said~~ the packet.

5. (Currently amended) The method of claim 1, further comprising:

recording a stuffing time difference in a service header at ~~said~~ the ingress end; and implementing ~~said~~ the stuffing time difference at ~~said~~ the egress end.

6. (Currently amended) The method of claim 1, wherein ~~said~~ the service header includes a structure pointer to indicate whether a header byte indicating a start of a synchronous payload envelope is present within a packet, ~~said~~ the structure pointer indicating a location of ~~said~~ the header byte in ~~said~~ the packet.

7. (Currently amended) The method of claim 1, further comprising:

checking a sequence counter in ~~said~~ the service header of each packet in ~~said~~ the set of packets;

locating at least one header byte in ~~said~~ the set of packets;

measuring all bytes between two header bytes; and

pushing ~~said~~ the set of packets into a frame.

8. (Currently amended) The method of claim 1, further comprising:

checking a sequence counter in ~~said~~ the service header of each packet in ~~said~~ the set of packets to determine if all packets are present sequentially: and

inserting a dummy packet if a packet is missing in ~~said~~ the set of packets.

9. (Currently amended) The method of claim 8, further comprising:

receiving an out of sequence packet; and

discarding ~~said~~ the out of sequence packet.

10. (Currently amended) The method of claim 1, further comprising:

checking a sequence counter in ~~said~~ the service header of each packet in ~~said~~ the set of packets to determine if all packets are present sequentially;

terminating a current connection if multiple packets are missing in ~~said~~ the set of packets;

discarding ~~said~~ the set of packets; and

establishing a new connection to begin receiving packets.

11. (Currently amended) The method of claim 1, further comprising:

checking a sequence counter in ~~said~~ the service header of each packet in ~~said~~ the set of packets to determine if all packets are present sequentially; and

establishing an in-frame condition after ~~said~~ the set of packets are received in sequence.

12. (Currently amended) The method of claim 11, further comprising:

determining whether ~~said~~ the in-frame condition is valid; and

terminating a current connection if ~~said~~ the in-frame condition is not valid.

13. (Currently amended) A computer readable storage medium including code for circuit emulation over a multi-packet label switching (MPLS) network, the code operable to:

receive a time division multiplexed data stream at an ingress end from an optical carrier;

divide ~~said~~ the data stream into a set of fixed sized packets;

add a service header to each of ~~said~~ the packets;

add an additional header on top of ~~said~~ the service header in accordance with MPLS protocols;

remove ~~said~~ the additional header after each packet has been processed by ~~said~~ the MPLS network;

use ~~said~~ the service header to recover ~~said~~ the data stream at an egress end;

store a first set of frames of the data stream into a data buffer during times clocks for the ingress and egress ends cannot be traced to a common reference source;

calculate a first data average of ~~said~~ the first set of frames in ~~said~~ the data buffer to obtain a threshold value;

store a next set of frames into ~~said~~ the data buffer;

calculate a next data average of ~~said~~ the next set of frames in ~~said~~ the data buffer;

compare ~~said~~ the next data average to ~~said~~ the threshold value; and

if ~~said~~ the next data average is greater than ~~said~~ the threshold value[[;]], generate a negative justification indicator[[;]] and send one more byte at ~~said~~ the egress end[[;]], or if ~~said~~ the next data average is less than ~~said~~ the threshold value[[;]], generate a positive justification indicator[[;]] and send one less byte at ~~said~~ the egress end.

14. (Currently amended) The computer readable storage medium of claim 13, wherein the code is further operable to:

monitor ~~said~~ the data stream; and

attach an alarm bit in a service header of a subsequent packet if a break in ~~said~~ the data stream is detected.

15. (Currently amended) The computer readable storage medium of claim 13, wherein the code is further operable to:

use a negative justification bit and a positive justification bit in ~~said~~ the service header to indicate whether ~~said~~ the synchronous payload envelope includes a negative stuff byte or a positive stuff byte.

16. (Currently amended) The computer readable storage medium of claim [[13]] 18 wherein the ~~code is further operable to: reserve~~ structure pointer reserves a pointer value indicating that ~~said~~ the header byte is not present within ~~said~~ the packet.

17. (Currently amended) The computer readable storage medium of claim 13, wherein the code is further operable to:

record a stuffing time difference in a service header at ~~said~~ the ingress end; and implement ~~said~~ the stuffing time difference at ~~said~~ the egress end.

18. (Currently amended) The computer readable storage medium of claim 13, wherein ~~said~~ the service header includes a structure pointer to indicate whether a header byte indicating a start of a synchronous payload envelope is present within a packet, ~~said~~ the structure pointer indicating a location of ~~said~~ the header byte in ~~said~~ the packet.

19. (Currently amended) The computer readable storage medium of claim 13, wherein the code is further operable to:

check a sequence counter in ~~said~~ the service header of each packet in ~~said~~ the set of packets;

locate at least one header byte in ~~said~~ the set of packets;

measure all bytes between two header bytes; and

push ~~said~~ the set of packets into a frame.

20. (Currently amended) The computer readable storage medium of claim 13, wherein the code is further operable to:

check a sequence counter in ~~said~~ the service header of each packet in ~~said~~ the set of packets to determine if all packets are present sequentially; and

insert a dummy packet if a packet is missing in ~~said~~ the set of packets.

21. (Currently amended) The computer readable storage medium of claim 20, wherein the code is further operable to:

receive an out of sequence packet; and

logic code for discarding ~~said~~ the out of sequence packet.

22. (Currently amended) The computer readable storage medium of claim 13, wherein the code is further operable to:

check a sequence counter in ~~said~~ the service header of each packet in ~~said~~ the set of packets to determine if all packets are present sequentially;

establish an in-frame condition after all packets for a frame are received in sequence;

terminate a current connection if multiple packets are missing in ~~said~~ the set of packets;

discard ~~said~~ the set of packets; and

establish a new connection to begin receiving packets.

23. (Currently amended) The computer readable storage medium of claim 22, wherein the code is further operable to:

check a sequence counter in ~~said~~ the service header of each packet in ~~said~~ the set of packets to determine if all packets are present sequentially;

establish an in-frame condition after all packets for a frame are received in sequence.

24. (Currently amended) The computer readable storage medium of claim 23, wherein the code is further operable to:

determine whether ~~said~~ the in-frame condition is valid; and

terminate a current connection if ~~said~~ the in-frame condition is not valid.